# 日本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２０００年１２月１１日

出願番号
Application Number: 特願２０００－３７５６４１

出願人
Applicant (s): 松下電器産業株式会社

２００１年　３月３０日

特許庁長官
Commissioner,
Patent Office 及川耕造

出証番号　出証特２００１－３０２５５４７

【書類名】　特許願

【整理番号】　2022520408

【提出日】　平成12年12月11日

【あて先】　特許庁長官殿

【国際特許分類】　G06F 11/28

【発明者】

【住所又は居所】　東広島市鏡山３丁目１０番１８号　株式会社松下電器情報システム広島研究所内

【氏名】　三宅　秀明

【発明者】

【住所又は居所】　東広島市鏡山３丁目１０番１８号　株式会社松下電器情報システム広島研究所内

【氏名】　枯木　　正吉

【発明者】

【住所又は居所】　大阪府門真市大字門真１００６番地　　松下電器産業株式会社内

【氏名】　鶴本　　克己

【特許出願人】

【識別番号】　000005821

【氏名又は名称】　松下電器産業株式会社

【代理人】

【識別番号】　100097445

【弁理士】

【氏名又は名称】　岩橋　文雄

【選任した代理人】

【識別番号】　100103355

【弁理士】

【氏名又は名称】　坂口　智康

【選任した代理人】

【識別番号】　100109667

【弁理士】

【氏名又は名称】　内藤　浩樹

【先の出願に基づく優先権主張】

【出願番号】　特願2000- 94064

【出願日】　平成12年 3月30日

【手数料の表示】

【予納台帳番号】　011305

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9809938

【書類名】　　明細書

【発明の名称】　デバッグ支援装置、デバッグ支援方法及びそのプログラムを記録したコンピュータ読み取り可能な記録媒体

【特許請求の範囲】

【請求項１】　汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、

前記ＯＳシミュレータで変更され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する記憶手段と、

前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、

前記アプリケーションと前記ＯＳデバッガとで共有され、前記記憶手段に記憶されたＯＳ管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルと、

前記記憶手段のＯＳ管理情報を前記共有ファイルの共有管理情報に書き込む書込手段と、

前記書込手段でＯＳ管理情報が書き込まれた共有管理情報を読み出す読出手段とを備えることを特徴とするデバッグ支援装置。

【請求項２】　前記書込手段は、前記アプリケーション側に備えられ、前記ＯＳシミュレータの要求に応じて前記ＯＳ管理情報の変更を前記共有管理情報に書き込み、

前記読出手段は、前記ＯＳデバッガ側に備えられ、当該ＯＳデバッガの要求に応じて前記書込手段でＯＳ管理情報が書き込まれた共有管理情報を読み出すことを特徴とする請求項１に記載のデバッグ支援装置。

【請求項３】　前記読出手段は、任意のタイミングで前記共有ファイルから共有管理情報を読み出すことを特徴とする請求項２に記載のデバッグ支援装置。

【請求項４】　前記読出手段は、所定の周期で前記共有ファイルから共有管理情報を読み出すことを特徴とする請求項２に記載のデバッグ支援装置。

【請求項５】　汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミ

ュレータがリンクされたアプリケーションと、

前記ＯＳシミュレータで変更され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する記憶手段と、

前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、

前記アプリケーションと前記ＯＳデバッガとで共有され、前記記憶手段に記憶されたＯＳ管理情報を変更するためのデバッグ命令を含む共有管理情報が記憶される共有ファイルと、

外部からの指示に応じたデバッグ命令を前記共有ファイルの共有管理情報に書き込むデバッグ命令書込手段と、

前記共有ファイルに記憶された共有管理情報から前記デバッグ命令書込手段で書き込まれたデバッグ命令を読み出すデバッグ命令読出手段と、

前記デバッグ命令読出手段で読み出されたデバッグ命令に対応して前記ＯＳ管理情報を変更する変更手段と

を備えることを特徴とするデバッグ支援装置。

【請求項６】　前記デバッグ命令書込手段は、前記ＯＳデバッガ側に備えられ、当該ＯＳデバッガの要求に応じてデバッグ命令を前記共有管理情報に書き込み、

前記デバッグ命令読出手段は、前記アプリケーション側に備えられ、前記デバッグ命令書込手段で書き込まれたデバッグ命令を読み出すことを特徴とする請求項５に記載のデバッグ支援装置。

【請求項７】　前記デバッグ命令読出手段は、任意のタイミングで前記共有ファイルからデバッグ命令を読み出す請求項５または６に記載のデバッグ支援装置。

【請求項８】　前記デバッグ命令読出手段は、所定の周期で前記共有ファイルからデバッグ命令を読み出す請求項５または６に記載のデバッグ支援装置。

【請求項９】　汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、

前記ＯＳシミュレータで変更され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する記憶手段と、

前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、

前記デバッグ支援装置は、

前記アプリケーションと前記ＯＳデバッガとで共有され、前記記憶手段に記憶されたＯＳ管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルと、

前記ＯＳシミュレータによって前記記憶手段のＯＳ管理情報が変更された場合、当該変更されたＯＳ管理情報を前記共有ファイルの共有管理情報に書き込む書込手段と、

前記アプリケーションと前記ＯＳデバッガとの間でプロセス間通信を実行する通信手段と、

前記共有ファイルに記憶された共有管理情報を読み出す読出手段とを更に備えており、

前記通信手段は、

前記ＯＳ管理情報の変更が前記共有管理情報に前記書込手段によって書き込まれた場合、前記共有管理情報を読み出す指示を前記アプリケーションから前記ＯＳデバッガに送信して前記アプリケーションの実行を停止させ、

前記共有管理情報が前記読出手段によって読み出された場合、前記アプリケーションを再起動する指示を前記ＯＳデバッガから前記アプリケーションに返信して、当該アプリケーションの実行が再開されることを特徴とするデバッグ支援装置。

【請求項１０】　前記通信手段は、前記アプリケーション側、前記ＯＳデバッガ側それぞれに備えられており、

前記アプリケーションの通信手段は、前記ＯＳ管理情報の変更が前記共有管理情報に前記書込手段によって書き込まれた場合、前記共有管理情報を読み出す指示を前記ＯＳデバッガの通信手段に送信し、

前記ＯＳデバッガの通信手段は、前記共有管理情報が前記読出手段によって読み出された場合、前記アプリケーションを再起動する指示を前記アプリケーションの通信手段に返信することを特徴とする請求項９に記載のデバッグ支援装置。

【請求項１１】　汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、

前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、

前記アプリケーションと前記ＯＳデバッガとで共有され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する共有ファイルと、

前記ＯＳシミュレータの要求に応じて、前記共有ファイルに記憶されたＯＳ管理情報を更新する更新手段と、

前記ＯＳデバッガの要求に応じて、前記更新手段で更新されたＯＳ管理情報を読み出す読出手段と

を備えることを特徴とするデバッグ支援装置。

【請求項１２】　汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、

前記ＯＳシミュレータで変更され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する記憶手段と、

前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有するコンピュータにおいて、前記アプリケーションをデバッグするデバッグ支援方法であって、

前記コンピュータは、

前記アプリケーションと前記ＯＳデバッガとで共有され、前記記憶手段に記憶されたＯＳ管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルを更に有し、

前記記憶手段のＯＳ管理情報を前記共有ファイルの共有管理情報に書き込む書込ステップと、

前記書込ステップでＯＳ管理情報が書き込まれた共有管理情報を読み出す読出

ステップと

を含むことを特徴とするデバッグ支援方法。

【請求項１３】　汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、

前記ＯＳシミュレータで変更され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する記憶手段と、

前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有するコンピュータにおいて、前記アプリケーションをデバッグするデバッグ支援方法であって、

前記コンピュータは、

前記アプリケーションと前記ＯＳデバッガとで共有され、前記記憶手段に記憶されたＯＳ管理情報を変更するためのデバッグ命令を含む共有管理情報が記憶される共有ファイルを更に有しており、

外部からの指示に応じたデバッグ命令を前記共有ファイルの共有管理情報に書き込むデバッグ命令書込ステップと、

前記共有ファイルに記憶された共有管理情報から前記デバッグ命令書込ステップで書き込まれたデバッグ命令を読み出すデバッグ命令読出ステップと、

前記デバッグ命令読出ステップで読み出されたデバッグ命令に対応して前記ＯＳ管理情報を変更する変更ステップと

を含むことを特徴とするデバッグ支援方法。

【請求項１４】　汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、

前記ＯＳシミュレータで変更され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する記憶手段と、

前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有するコンピュータにおいて、前記アプリケーションをデバッグするデバッグ支援方法であって、

前記コンピュータは、

前記アプリケーションと前記ＯＳデバッガとで共有され、前記記憶手段に記憶さ

れたＯＳ管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルを更に有し、

前記ＯＳシミュレータによって前記記憶手段のＯＳ管理情報が変更された場合、当該変更されたＯＳ管理情報を前記共有ファイルの共有管理情報に書き込む書込ステップと、

前記アプリケーションと前記ＯＳデバッガとの間でプロセス間通信を実行する通信ステップと、

前記共有ファイルに記憶された共有管理情報を読み出す読出ステップとを含み、

前記通信ステップでは、

前記ＯＳ管理情報の変更が前記共有管理情報に前記書込ステップによって書き込まれた場合、前記共有管理情報を読み出す指示を前記アプリケーションから前記ＯＳデバッガに送信して前記アプリケーションの実行を停止させ、

前記共有管理情報が前記読出ステップで読み出された場合、前記アプリケーションを再起動する指示を前記ＯＳデバッガから前記アプリケーションに返信して、当該アプリケーションの実行が再開されることを特徴とするデバッグ支援方法。

【請求項１５】　前記アプリケーション、ＯＳデバッガはそれぞれ通信手段を備えており、

前記通信ステップでは、

前記ＯＳ管理情報の変更が前記共有管理情報に前記書込手段によって書き込まれた場合、前記アプリケーションの通信手段によって前記共有管理情報を読み出す指示を前記ＯＳデバッガの通信手段に送信し、

前記共有管理情報が前記読出ステップによって読み出された場合、前記ＯＳデバッガの通信手段によって前記アプリケーションを再起動する指示を前記アプリケーションの通信手段に返信することを特徴とする請求項１４に記載のデバッグ支援方法。

【請求項１６】　汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、

前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデ

バッガとを有するコンピュータにおいて、前記アプリケーションをデバッグするデバッグ支援方法であって、

前記コンピュータは、

前記アプリケーションと前記ＯＳデバッガとで共有され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する共有ファイルを有し、

前記ＯＳシミュレータの要求に応じて、前記共有ファイルに記憶されたＯＳ管理情報を更新する更新ステップと、

前記ＯＳデバッガの要求に応じて、前記更新ステップで更新されたＯＳ管理情報を読み出す読出ステップと

を含むことを特徴とするデバッグ支援方法。

【請求項１７】　汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、

前記ＯＳシミュレータで変更され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する記憶手段と、

前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有するコンピュータで実行され、前記アプリケーションをデバッグするプログラムが記録されたコンピュータ読み取り可能な記録媒体であって、

前記コンピュータは、

前記アプリケーションと前記ＯＳデバッガとで共有され、前記記憶手段に記憶されたＯＳ管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルを更に有しており、

前記コンピュータに、

前記記憶手段のＯＳ管理情報を前記共有ファイルの共有管理情報に書き込む書込ステップと、

前記書込ステップでＯＳ管理情報が書き込まれた共有管理情報を読み出す読出ステップと

を実行させるプログラムが記録された記録媒体。

【請求項１８】　汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、

・前記ＯＳシミュレータで変更され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する記憶手段と、

前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有するコンピュータで実行され、前記アプリケーションをデバッグするプログラムが記録されたコンピュータ読み取り可能な記録媒体であって、

前記コンピュータは、

前記アプリケーションと前記ＯＳデバッガとで共有され、前記記憶手段に記憶されたＯＳ管理情報を変更するためのデバッグ命令を含む共有管理情報が記憶される共有ファイルを更に有しており、

前記コンピュータに、

外部からの指示に応じたデバッグ命令を前記共有ファイルの共有管理情報に書き込むデバッグ命令書込ステップと、

前記共有ファイルに記憶された共有管理情報から前記デバッグ命令書込ステップで書き込まれたデバッグ命令を読み出すデバッグ命令読出ステップと、

前記デバッグ命令読出ステップで読み出されたデバッグ命令に対応して前記ＯＳ管理情報を変更する変更ステップと

を実行させるプログラムが記録された記録媒体。

【請求項１９】　汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、

前記ＯＳシミュレータで変更され、当該アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する記憶手段と、

前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有するコンピュータで実行され、前記アプリケーションをデバッグするプログラムが記録されたコンピュータ読み取り可能な記録媒体であって、

前記コンピュータは、

前記アプリケーションと前記ＯＳデバッガとで共有され、前記記憶手段に記憶されたＯＳ管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルを更に有しており、

前記コンピュータに、

前記ＯＳシミュレータによって前記記憶手段のＯＳ管理情報が変更された場合、当該変更されたＯＳ管理情報を前記共有ファイルの共有管理情報に書き込む書込ステップと、

前記アプリケーションと前記ＯＳデバッガとの間でプロセス間通信を実行する通信ステップと、

前記共有ファイルに記憶された共有管理情報を読み出す読出ステップとを実行させ、

前記通信ステップでは、

前記ＯＳ管理情報の変更が前記共有管理情報に前記書込ステップによって書き込まれた場合、前記共有管理情報を読み出す指示を前記アプリケーションから前記ＯＳデバッガに送信して前記アプリケーションの実行を停止させ、

前記共有管理情報が前記読出ステップで読み出された場合、前記アプリケーションを再起動する指示を前記ＯＳデバッガから前記アプリケーションに返信して、当該アプリケーションの実行が再開されることを特徴とする記録媒体。

【請求項２０】　前記アプリケーション、ＯＳデバッガはそれぞれ通信手段を備えており、

前記通信ステップでは、

前記ＯＳ管理情報の変更が前記共有管理情報に前記書込手段によって書き込まれた場合、前記アプリケーションの通信手段によって前記共有管理情報を読み出す指示を前記ＯＳデバッガの通信手段に送信し、

前記共有管理情報が前記読出ステップによって読み出された場合、前記ＯＳデバッガの通信手段によって前記アプリケーションを再起動する指示を前記アプリケーションの通信手段に返信することを特徴とする請求項１９に記載の記録媒体。

【請求項２１】　汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、

前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有するコンピュータで実行され、前記アプリケーションをデバッグするプログラムが記録されたコンピュータ読み取り可能な記録媒体であって、

前記コンピュータは、

・前記アプリケーションと前記ＯＳデバッガとで共有され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する共有ファイルを有し、

前記コンピュータに、

前記ＯＳシミュレータの要求に応じて、前記共有ファイルに記憶されたＯＳ管理情報を更新する更新ステップと、

前記ＯＳデバッガの要求に応じて、前記更新ステップで更新されたＯＳ管理情報を読み出す読出ステップと

を実行させるプログラムが記録された記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、デバッグ支援装置、デバッグ支援方法及びそのプログラムを記録した記録媒体に関し、特に、ＯＳシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際に用いるデバッグ支援装置に関するものである。

【０００２】

【従来の技術】

近年急速に普及しつつある携帯電話などの移動体端末（以下、機器）では、通話機能、電話帳登録機能、インターネット通信など様々な機能の実現が要求されている。そのため移動体端末には、このような様々な機能を実現するためのアプリケーションが組み込まれるようになっている。

【０００３】

図７は、組み込み型のアプリケーションの開発環境の説明図である。図７（ａ）は、機器にアプリケーションが組み込まれた状態を表す概念図である。アプリケーション２１０は、通常、機器専用のオペレーティングシステム（以下、専用ＯＳ；Operating System）２２０ａ上で動作するようになっている。アプリケーション２１０からの要求にしたがって、専用ＯＳ２２０ａが、機器内のハードウェアを制御することによって様々な機能が実現されることになる。

【０００４】

・従って、このような組み込み型のアプリケーション２１０の開発は機器のハードウェアや専用ＯＳ２０ａの開発と並行して進める必要がある。通常、アプリケーション２１０の組み込み先である機器の環境を想定した仮想的な環境（以下、シミュレーション環境）が構築された汎用コンピュータ上で、アプリケーション２１０の開発が汎用ＯＳを用いて行われる。

【０００５】

図７（ｂ）は、汎用コンピュータ上のシミュレーション環境の状態を表す説明図である。汎用コンピュータ内のハードウェアを制御する汎用ＯＳ２５０上で、アプリケーション２１０、ＯＳシミュレータ２２０、アプリケーションデバッガ２３０、ＯＳデバッガ２４０が実行される。図７（ｂ）では、専用ＯＳ２２０ａの動作（例えば、タスク管理動作や、イベントフラグなどのリソース管理動作）を仮想的に実行するＯＳシミュレータ２２０が、ライブラリという形式でアプリケーション１０にリンクされている。これにより、アプリケーション２１０が組み込まれる機器のシミュレーション環境が構築されている。

【０００６】

また、アプリケーション２１０のデバッグは、汎用のアプリケーション開発ツール（図示せず）に含まれるアプリケーションデバッガ２３０を用いて行うようになっている。ここでデバッグとは、プログラム中のバグを取り除く作業をいい、デバッガとは、デバッグを効率よく行えるようにしたユーティリティプログラムをいう。

【０００７】

通常、アプリケーション２１０をデバッグする場合、図示しないメモリにおいてＯＳシミュレータ２２０が管理する各種情報（以下、ＯＳ管理情報）を参照あるいは変更する場合がある。ＯＳ管理情報には、例えば、アプリケーション２１０の実行中におけるタスクの状態や、タスクの状態を変化させるイベントの有無を１または０のビットパターンで表したイベントフラグなどがある。

【０００８】

しかしながら、アプリケーションデバッガ２３０は、汎用のアプリケーション

開発ツールであるため、ＯＳ管理情報を参照あるいは変更するというような機能は備えていない。そこで、最近では、アプリケーションデバッガ２３０だけでなく、ＯＳ管理情報を操作するためのユーティリティプログラムであるＯＳデバッガ２４０を備えたデバッグ支援システムが登場している。

【０００９】

図６は、従来におけるデバッグ支援システムの一例を示す機能ブロック図である。図６に示すように、ＯＳデバッガ２４０は、ＯＳシミュレータ２２０とは別のプロセス空間に存在するため、ＯＳ管理情報に直接アクセスすることはできない。従って、アプリケーション２１０側とＯＳデバッガ２４０側の両方に通信手段Ｃ１、Ｃ２ を備え、通信手段Ｃ１ 、Ｃ２ でプロセス間通信を行うことによって、ＯＳデバッガ２４０からＯＳ管理情報を操作することが可能となる。

【００１０】

また、デバッグ作業時においてバグの位置を見当付けるのを容易にするため、ブレークポイントを設定するなどしてアプリケーション２１０を停止させ、この時点のＯＳ管理情報を参照することがある。

【００１１】

【発明が解決しようとする課題】

しかしながら、アプリケーション２１０が停止した場合は、アプリケーション２１０とリンクしているＯＳシミュレータ２２０も停止する。すなわち、図６に示したような従来のデバッグ支援システムのようにプロセス間通信を採用した構成によれば、アプリケーション２１０が停止すると、ＯＳシミュレータ２２０とＯＳデバッガ２４０との間で通信が行えなくなり、この時点のＯＳ管理情報を参照できなくなり、デバッグ作業時においてバグの位置を見当付けるのが困難であった。

【００１２】

ブレークポイントを設定してアプリケーション２１０が意識的に停止された場合だけでなく、アプリケーション２１０がハングアップした場合もＯＳ管理情報を参照できず、ブレークポイントが設定された場合よりも更にバグの位置を見当付けることが困難となる。

【００１３】

本発明の目的は、ＯＳシミュレータによるシミュレーション環境で開発したアプリケーションが停止した場合でも、この時点のＯＳ管理情報を確実に参照でき、アプリケーションのデバッグの効率を向上させることである。

【００１４】

【課題を解決するための手段】

上記課題を解決するために、本発明の第１の態様によれば、汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、前記ＯＳシミュレータで変更され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する記憶手段と、前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、前記アプリケーションと前記ＯＳデバッガとで共有され、前記記憶手段に記憶されたＯＳ管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルと、前記記憶手段のＯＳ管理情報を前記共有ファイルの共有管理情報に書き込む書込手段と、 前記書込手段でＯＳ管理情報が書き込まれた共有管理情報を読み出す読出手段とを備えるようになっている。

【００１５】

また、上記課題を解決するために、本発明の第２の態様によれば、汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、前記ＯＳシミュレータで変更され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する記憶手段と、前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、前記アプリケーションと前記ＯＳデバッガとで共有され、前記記憶手段に記憶されたＯＳ管理情報を変更するためのデバッグ命令を含む共有管理情報が記憶される共有ファイルと、外部からの指示に応じたデバッグ命令を前記共有ファイルの共有管理情報に書き込むデバッグ命令書込手段と、前記共有ファイルに記憶された共有管理情報から前記デバッグ命令書込手段で書き込まれたデバッグ命令を読み出すデバッグ命令読

出手段と、前記デバッグ命令読出手段で読み出されたデバッグ命令に対応して前記ＯＳ管理情報を変更する変更手段とを備えるようになっている。

【００１６】

また、上記課題を解決するために、本発明の第３の態様によれば、汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、前記ＯＳシミュレータで変更され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する記憶手段と、前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、

前記デバッグ支援装置は、前記アプリケーションと前記ＯＳデバッガとで共有され、前記記憶手段に記憶されたＯＳ管理情報と同一のデータを含む共有管理情報が記憶される共有ファイルと、前記ＯＳシミュレータによって前記記憶手段のＯＳ管理情報が変更された場合、当該変更されたＯＳ管理情報を前記共有ファイルの共有管理情報に書き込む書込手段と、前記アプリケーションと前記ＯＳデバッガとの間でプロセス間通信を実行する通信手段と、前記共有ファイルに記憶された共有管理情報を読み出す読出手段とを更に備えており、前記通信手段は、前記ＯＳ管理情報の変更が前記共有管理情報に前記書込手段によって書き込まれた場合、前記共有管理情報を読み出す指示を前記アプリケーションから前記ＯＳデバッガに送信して前記アプリケーションの実行を停止させ、前記共有管理情報が前記読出手段によって読み出された場合、前記アプリケーションを再起動する指示を前記ＯＳデバッガから前記アプリケーションに返信して、当該アプリケーションの実行が再開されるようになっている。

【００１７】

また、上記課題を解決するために、本発明の第４の態様によれば、汎用ＯＳ上でリアルタイムＯＳをシミュレートするＯＳシミュレータがリンクされたアプリケーションと、前記ＯＳシミュレータで変更されるＯＳ管理情報を参照または変更するＯＳデバッガとを有し、前記アプリケーションをデバッグするデバッグ支援装置であって、前記アプリケーションと前記ＯＳデバッガとで共有され、前記アプリケーションのタスク実行を制御するＯＳ管理情報を記憶する共有ファイル

と、前記ＯＳシミュレータの要求に応じて、前記共有ファイルに記憶されたＯＳ管理情報を更新する更新手段と、前記ＯＳデバッガの要求に応じて、前記更新手段で更新されたＯＳ管理情報を読み出す読出手段とを備えるようになっている。

【００１８】

【発明の実施の形態】

以下に本発明の一実施の形態を図面に従って詳細に説明する。

（第１の実施の形態）

以下、本発明のデバック支援装置における第１の実施の形態について図面を参照しながら詳細に説明する。本実施の形態においては、本発明におけるデバッグ支援装置が携帯電話などの移動体端末のアプリケーションのデバッグ作業に適用された場合について説明する。

【００１９】

図１は、本発明を適用したデバッグ支援装置の機能ブロック図である。本実施の形態に係るデバッグ支援装置１１０は、例えば、アプリケーション１０、ＯＳシミュレータ２０、記憶手段９０、ＯＳデバッガ４０、共有ファイルＦ、表示制御部１２１、インターフェイス部１２２及び図示しないアプリケーションデバッガとを備えている。本デバッグ支援装置１１０においては、汎用ＯＳ（図示せず）、ＯＳシミュレータ２０及びＯＳデバッガ４０が協働して、アプリケーション１０が組み込まれる機器のシミュレーション環境が構築されている。

【００２０】

アプリケーション１０は、移動体端末の様々な業務処理を行うためのプログラムである。アプリケーション１０は、ライブラリという形式でＯＳシミュレータ２０がリンクされており、共有管理情報書き込み手段１２を備えている。

【００２１】

ＯＳシミュレータ２０は、システムコール処理手段２１とＯＳ管理情報変更手段２２とを備えている。ＯＳシミュレータ２０は、タスク管理やイベントフラグなどのリソース管理といった専用ＯＳ（図示せず）の動作を擬似的に実行する。システムコール処理手段２１は、アプリケーションデバッガ（図示せず）によってアプリケーション１０のタスクが実行されたときに発行されるシステムコール

を処理し、システムコールの処理によるＯＳ管理情報を変更する命令をＯＳ管理情報変更手段２２に通知する。ここで、ＯＳ管理情報とは、ＯＳシミュレータが管理するデータであり、例えばアプリケーション１０実行中のタスクの状態をあらわすタスク実行状況やタスクの状態を変化させるイベントの有無を１又は０のビットパターンで表したイベントフラグなどがある。

【００２２】

ＯＳ管理情報変更手段２２は、システムコール処理手段２１からのＯＳ管理情報の変更命令に従って、後述する記憶手段９０に格納されているＯＳ管理情報ＯＳ２を変更し、更にＯＳ管理情報ＯＳ２の変更内容を共有管理情報書込み手段１２に通知する。

【００２３】

共有管理情報書き込み手段１２は、ＯＳ管理情報変更手段２２から通知されたＯＳ管理情報ＯＳ２の変更内容を後述する共有ファイルＦに書き込む。

【００２４】

記憶手段９０は、ＲＡＭ（Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ）で構成される。記憶手段９０には、ＯＳ管理情報変更手段２２で変更されるＯＳ管理情報ＯＳ２が記憶される。図１においては、一例として、アプリケーション１０のタスクＡに対応するＯＳ管理情報が記憶されている。ＯＳ管理情報ＯＳ２には、イベントフラグＥＦ２が含まれている。イベントフラグＥＦ２は、タスクＡの状態を変化させるイベントの有無を１又は０のビットパターンで表したものである。

【００２５】

共有ファイルＦは、アプリケーション１０やＯＳデバッガ４０によって共有されるハードディスク内のメモリ領域に記憶され、各プロセスで共有される共有管理情報ＳＳ２を含むデータ群を示す。共有ファイルＦには、記憶手段９０に記憶されたＯＳ管理情報と同一のデータが含まれたデータである共有管理情報ＳＳ２が格納される。アプリケーション１０のタスク毎の実行に伴う、ＯＳシミュレータ２０によるＯＳ管理情報ＯＳ２の変更に同期して、ＯＳ管理情報ＯＳ２の変更が共有ファイルＦの共有管理情報ＳＳ２に反映されるようになっている。これに

より、アプリケーション１０、ＯＳデバッガ４０それぞれのプロセスで、共有ファイルＦを介してＯＳ管理情報ＯＳ２の共有化が実現できる。また、共有管理情報ＳＳ２はアプリケーション１０とＯＳデバッガ４０とで共有されるので、ハードディスクなどの不揮発性メモリ上に記憶することが望ましい。ＦＤなどの外部記録媒体であってもよい。共有管理情報ＳＳ２が消去されることがなく、アプリケーション１０とＯＳデバッガ４０とでデータの共有化が確実にできるためである。

【００２６】

図１においては、一例として、アプリケーション１０のタスクＡに対応するＯＳ管理情報と同一のデータを含む共有管理情報ＳＳ２が記憶されている。共有管理情報ＳＳ２には、イベントフラグＥＦ３が含まれており、イベントフラグＥＦ３は記憶手段９０のイベントフラグＥＦ２に対応する。つまり、記憶手段９０のＯＳ管理情報ＯＳ２と共有ファイルＦの共有管理情報ＳＳ１とは完全に同期されるようになっている。

【００２７】

ＯＳ管理情報ＯＳ２を操作するためのユーティリティプログラムであるＯＳデバッガ４０は、ユーザインターフェイス４１、メイン処理手段４２、共有管理情報読み出し手段４３、表示処理手段４４とを備えている。

【００２８】

共有管理情報読み出し手段４３は、メイン処理手段４２の命令により、共有管理情報ＳＳ２を読み出し、読み出された共有管理情報ＳＳ２を表示するよう表示処理手段４４へ命令する。表示処理手段４４は、共有管理情報読み出し手段４３によって読み出された共有管理情報ＳＳ２を表示制御部１２１で映像信号に変換し、ディスプレイの画面（図示せず）に表示する。

【００２９】

ユーザインターフェイス４１は、ユーザによって操作部（図示せず）からの入力信号がインターフェイス部１２２を介して通知された場合、共有ファイルＦ１に格納された共有管理情報ＳＳ２を表示させる処理をメイン処理手段４２へ命令する。メイン処理手段４２は、ユーザインターフェイス４１の命令により、共有

管理情報読み出し手段４３へ共有管理情報ＳＳ２を読み出す命令をする。

【００３０】

ここで、本実施の形態では、図示しない中央演算処理装置（ＣＰＵ）を含む制御部が、本発明のプログラムにしたがってＣＰＵ以外の各回路と協働して実行される制御動作を実現している。しかし、以下では説明の便宜上、ＣＰＵが関係する制御動作を、アプリケーション１０などのプログラムが直接的に制御しているものとしてその説明を行う。

【００３１】

ここでは、例として、タスクＡが実行され、システムコールである「ｓｅｔ＿ｆｌｇ」、「ｃｌｒ＿ｆｌｇ」が発行される場合を用いて説明する。これらのシステムコールによってイベントフラグの任意のビットがセットされ、他のタスクの状態を待ち状態としたり、待ち状態を解除して実行可能状態と変更できる。システムコール「ｓｅｔ＿ｆｌｇ」が発行されるとＯＳ管理情報ＯＳ２のイベントフラグＥＦ２の値が０から１に変更される。また、システムコール「ｃｌｒ＿ｆｌｇ」が発行されるとＯＳ管理情報ＯＳ２のイベントフラグＥＦ２の値が１から０に変更されるようになっている。

【００３２】

まず、アプリケーション１０側の処理の流れを説明する。アプリケーションデバッガ（図示せず）によってアプリケーション１０のタスクＡが実行されて、システムコール「ｓｅｔ＿ｆｌｇ」が発行される。発行されたシステムコール「ｓｅｔ＿ｆｌｇ」がシステムコール処理手段２１で処理され、ＯＳ管理情報ＯＳ２のイベントフラグＥＦ２の値を０から１に変更する命令がＯＳ管理情報変更手段２２に通知される。

【００３３】

続いて、ＯＳ管理情報変更手段２２は、ＯＳ管理情報ＯＳ２のイベントフラグＥＦ２の値を０から１に変更し、イベントフラグＥＦ２を０から１に変更した内容を共有管理情報書き込み手段１２に通知する。

【００３４】

共有管理情報書込み手段１２は、ＯＳ管理情報ＯＳ２の変更内容を共有ファイ

ルＦのイベントフラグＥＦ３に書き込む。具体的には、共有管理情報ＳＳ２のイベントフラグＥＦ３の値を０から１に変更する。

【００３５】

ＯＳ管理情報ＯＳ２の変更内容が共有ファイルＦに書き込まれた後、アプリケーション１０の動作が停止される。アプリケーション１０の動作が停止された時点のイベントフラグの状況を参照したい場合は、ユーザによってキーボードなどの操作部が操作され、操作部から入力された入力信号がインターフェイス部１２２を介してユーザインターフェイス４１に通知される。ユーザインターフェイス４１からの指示に応じて、メイン処理手段４２は、共有管理情報ＳＳ２のイベントフラグＥＦ３を読み出すように共有管理情報読み出し手段４３に命令する。

【００３６】

共有管理情報読み出し手段４３は、メイン処理手段４２からの命令に従って、共有管理情報ＳＳ２のイベントフラグＥＦ３を読み出す。読み出されたイベントフラグＥＦ３の情報は共有管理情報読み出し手段４３から表示処理手段４４へ通知され、表示制御部１２１で映像信号に変更されてディスプレイの画面（図示せず）に表示される。

【００３７】

この際、イベントフラグＥＦ３が１であることをユーザが視覚的に理解できるように画面上に表示させることによって、ユーザはアプリケーション１０と汎用ＯＳ（図示せず）との処理の連携、アプリケーション１０とＯＳデバッガ４０との処理間で同期がとられたことを容易に把握することができる。

【００３８】

以上のように、ＯＳ管理情報変更手段２２によるＯＳ管理情報ＯＳ２の変更に同期して、ＯＳ管理情報ＯＳ２の変更が共有ファイルＦの共有管理情報ＳＳ２に反映されるので、アプリケーション１０が停止した場合でもＯＳ管理情報ＯＳ２を容易に参照することができる。これにより、効率の良いデバッグが実行することができるようになる。

【００３９】

なお、イベントフラグの値を共有ファイルＦから読み出すタイミングは、「イ

ベントフラグの参照」が指示されたときに限定されるものではない。例えば、共有管理情報読み出し手段４３に対して周期的にイベントフラグＥＦ３を含む共有管理情報ＳＳ２の読み出しを指示する手段を備えた構成としてもよい。これにより、ＯＳ管理情報ＯＳ２の時間的遷移を知ることができるため、予期しないイレギュラーなアプリケーション１０の実行状態を容易に発見することができ、デバッグの効率が向上されることになる。

【００４０】

以上、システムコール「ｓｅｔ＿ｆｌｇ」の場合について説明したが、システムコール「ｃｌｒ＿ｆｌｇ」の場合でも同様にＯＳ管理情報ＯＳ１のイベントフラグＥＦ２が１から０に変更されたことをユーザに視覚的に伝えることができる。

【００４１】

また、ＯＳ管理情報ＯＳ２に含まれるデータの種別毎に共有ファイルＦ内のメモリ領域を予め割り当てておけば、イベントフラグだけでなく、他のＯＳ管理情報（タスクの状態・セマフォの状態・メールボックスの状態など）も同様の手順で参照できることはいうまでもない。

【００４２】

（第２の実施の形態）

以下、本発明のデバック支援装置における第２の実施の形態について図面を参照しながら詳細に説明する。本実施の形態においては、第１の実施の形態において、画面上で参照されたＯＳ管理情報ＯＳ２の値が不正である場合、ＯＳ管理情報ＯＳ２をＯＳデバッガ４０側から変更できるようになっている。

【００４３】

デバッグ中に不正な値を有するＯＳ管理情報ＯＳ２が発生した場合、アプリケーション１０にバグが存在するので、そのバグを検証して取り除く必要がある。しかし、一般的には、デバッグの効率を向上させるために、バグが発生するたびに個別に検証するのではなくアプリケーション１０の動作全体について検証が行われる。本デバッグ支援装置１２０では、画面上で参照されたＯＳ管理情報の値が不正である場合、ＯＳ管理情報ＯＳ２中の不正な値を有するデータをＯＳデバ

ッガ４０側から変更できる。これにより、アプリケーション１０のプログラムを修正することなく、アプリケーション１０の動作を継続させてデバッグの効率を向上させることが可能となる。

【００４４】

図２は、本実施の形態に係るデバッグ支援装置１２０の機能ブロック図である。本実施の形態に係るデバッグ支援装置１２０は、第１の実施の形態のデバッグ支援装置１１０と同様に、例えば、アプリケーション１０、ＯＳシミュレータ２０、記憶手段９０、ＯＳデバッガ４０、共有ファイルＦ、表示制御部１２１、インターフェイス部１２２及び図示しないアプリケーションデバッガとを備えている。本デバッグ支援装置１２０は、第１の実施の形態のデバッグ支援装置１１０と同様に、図示はしないが、アプリケーション１０側に共有管理情報書き込み手段１２、ＯＳデバッガ４０側に共有管理情報読み出し手段４３を備えており、共有ファイルＦへのデータの読み書き処理は可能である。以下、本実施の形態に係るデバッグ支援装置１２０と実施の形態１に係るデバッグ支援装置１１０との構成上の差異について説明する。

【００４５】

まず、アプリケーション１０は、ＯＳシミュレータ２０以外に、デバッグタスク起動手段１３、システムコール情報生成手段１４、デバッグ命令読み出し手段１５、タイマ１６を更に備えている。

【００４６】

デバッグ命令読み出し手段１５は、共有ファイルＦの共有管理情報ＳＳ２に含まれるデバッグ命令機能コードをタイマ１６を用いて周期的に読み出す。ここでデバッグ命令機能コードとは、ＯＳ管理情報ＯＳ２のデータをＯＳシミュレータ２０によって変更するためのシステムコールを特定するためのデータである。

【００４７】

システムコール情報生成手段１４は、デバッグ命令機能コードに対応するシステムコールの実行タスクを特定するための情報（以下、システムコール情報）を生成する。デバッグタスク起動手段１３は、システムコール情報生成手段１４で生成されたシステムコール情報に応じてデバッグタスク１７を起動する。起動さ

れたデバッグタスク１７がＯＳシミュレータ２０で実行処理されて、記憶手段９０のＯＳ管理情報ＯＳ２が変更されるようになっている。

【００４８】

また、ＯＳデバッガ４０は、ユーザインタフェース４１、メイン処理手段４２、表示処理手段４４以外に、デバッグ命令書き込み手段４５を更に備えている。

【００４９】

デバッグ命令書き込み手段４５は、ユーザからのＯＳ管理情報ＯＳ２の修正指示（以下、デバッグ命令）に応じて、デバッグ命令に対応するデバッグ命令機能コードを共有ファイルＦに書き込む。

【００５０】

以下、ＯＳデバッガ４０から、最終的にＯＳ管理情報ＯＳ２が変更されるまでの処理の流れを説明する。ここでは、第１の実施の形態で記載したように、タスクＡによってシステムコール「ｓｅｔ＿ｆｌｇ」が実行された後、参照された共有ファイルＦのイベントフラグＥＦ３が不正な値「０」であった場合について説明する。この場合、最終的にＯＳ管理情報ＯＳ２のイベントフラグＥＦ２を「１」に変更する必要がある。

【００５１】

まず、ＯＳデバッガ側からの処理の流れを説明する。イベントフラグＥＦ３が不正な値「０」であることがユーザによって確認された場合、ＯＳ管理情報ＯＳ２のイベントフラグＥＦ２を「１」に変更するためのデバッグ命令が操作部から入力される。入力されたデバッグ命令がインターフェイス部１２２を介してＯＳデバッガ４０のユーザインターフェイス４１に通知される。続いて、ユーザインターフェイス４１からの指示に応じて、メイン処理手段４２は、システムコール「ｓｅｔ＿ｆｌｇ」を特定するためのデバッグ命令機能コード（－４８）を共有管理情報ＳＳ２に書き込むようにデバッグ命令書き込み手段４５に命令する。

【００５２】

メイン処理手段４２からの命令に応じて、デバッグ命令書き込み手段４５によって、共有管理情報ＳＳ２中にデバッグ命令機能コードＣＯが書き込まれる。

【００５３】

・続いて、アプリケーション１０側の処理について説明する。デバッグ命令機能コードＣＯがデバッグ命令読み出し手段１５によって読み出され、システムコール情報生成手段１４に渡される。ここでデバッグ命令読み出し手段１５はタイマを用いて周期的に共有ファイルＦのデバッグ命令機能コードを読み出すが、読み出し周期はデバッグ命令機能コードが書き込まれる周期より短いのが望ましい。これにより、デバッグ命令機能コードの読み出しの取りこぼしを確実に防ぐことができるようになる。

【００５４】

デバッグ命令機能コードＣＯに対応する、例えばパラメータなどのシステムコール情報がシステムコール情報生成手段１４で生成され、デバッグタスク起動手段１３に渡される。渡されたシステムコール情報に従って、デバッグタスク起動手段１３は、システムコール「ｓｅｔ＿ｆｌｇ」を発行するデバッグタスク１７を起動する。

【００５５】

起動されたデバッグタスク１７によって、ＯＳシミュレータ２０に対してシステムコール「ｓｅｔ＿ｆｌｇ」が発行され、ＯＳ管理情報ＯＳ２のイベントフラグＥＦ２が「０」から「１」に変更される。イベントフラグＥＦ２の変更後、イベントフラグＥＦ２待ちの状態にあるタスクＢが実行状態に遷移される。

【００５６】

以上のように、本実施の形態によれば、何らかのバグが存在することによってＯＳ管理情報ＯＳ２が不正な値となった場合でも、このＯＳ管理情報ＯＳ２をＯＳデバッガ４０から変更することができる。また、このような変更機能を利用すれば、アプリケーション１０のプログラムを修正することなく、割り込みが発生した状況をつくることも可能である。

【００５７】

なお、システムコール情報を指定する方法は、システムコール「ｓｅｔ＿ｆｌｇ」を指示するパラメータだけでなく、システムコール「ｓｅｔ＿ｆｌｇ」を指示するメッセージの受け渡しであってもよい。また、共有ファイルＦを用いてシステムコール情報を指定してもよい。具体的には、デバッグタスク１７が直接、

共有ファイルＦの中のデバッグ命令機能コードを参照してシステムコール情報を生成するようにすればよい。

【００５８】

また、ＯＳデバッガ４０からデバッグ命令を指示することとしているが、本発明はこれに限定されるものではない。すなわち、予めデバッグ命令を共有ファイルＦに格納しておき、このデバッグ命令をデバッグ命令読み出し手段１５が周期的に（或いは、任意のタイミングで）読み出すようにしてもよい。このようにすれば、ＯＳデバッガ４０を用いることなくＯＳ管理情報ＯＳ２を変更できる。

【００５９】

また、デバッグ命令を実行する手段はデバッグタスク１７に限定されるものではない。すなわち、ここでは、デバッグタスク１７がシステムコールを発行することによってデバッグ命令が実行される動作を例示しているが、システムコールには、タスクからのみ発行可能・割り込みハンドラからのみ発行可能・この両方から発行可能という種別があるため、この種別に応じた実行手段を選択するようにしている。

【００６０】

更に、上記の説明では、デバッグタスク１７の状態遷移について特に言及していないが、システムコールを発行し終えたデバッグタスク１７は、消滅しても、常駐しても（すなわち、次のデバッグ命令が共有ファイルＦに書き込まれるまで待ち状態となっても）、いずれであってもかまわない。

（第３の実施の形態）

以下、本発明のデバック支援装置における第３の実施の形態について図面を参照しながら詳細に説明する。

【００６１】

本実施の形態におけるデバッグ支援装置は他の実施の形態と同様に共有ファイルを備えるとともに、更に、アプリケーションとＯＳデバッガとの間でプロセス間通信を実行する通信手段を備えていることを特徴とする。この通信手段により、ＯＳデバッガは、ＯＳシミュレータによってＯＳ管理情報が変更された場合に漏れなくＯＳ管理情報の変更内容を参照することができる。具体的には、ＯＳデ

バッガによってタイマを用いて共有ファイルからデータを周期的に読み出す場合、共有ファイルからデータを読み出す周期と、ＯＳ管理情報がＯＳシミュレータによって変更される周期を同期させることができる。

【００６２】

図３は、本実施の形態におけるデバッグ支援装置の機能ブロック図である。本実施の形態に係るデバッグ支援装置１は、例えば、アプリケーション６０、ＯＳシミュレータ７０、記憶手段１００、ＯＳデバッガ８０、共有ファイルＦ１、表示制御部９１、インターフェイス部９２及び図示しないアプリケーションデバッガとを備えている。本デバッグ支援装置１においては、汎用ＯＳ（図示せず）、ＯＳシミュレータ７０及びＯＳデバッガ８０が協働して、アプリケーション６０が組み込まれる機器のシミュレーション環境が構築されている。

【００６３】

アプリケーション６０は、移動体端末の様々な業務処理を行うためのプログラムである。アプリケーション６０は、ライブラリという形式でＯＳシミュレータ７０がリンクされており、共有管理情報書き込み手段６２とアプリケーション通信手段Ｃ１とを備えている。

【００６４】

ＯＳシミュレータ７０は、システムコール処理手段７１とＯＳ管理情報変更手段７２とを備えている。ＯＳシミュレータ７０は、タスク管理やイベントフラグなどのリソース管理といった専用ＯＳ（図示せず）の動作を擬似的に実行する。システムコール処理手段７１は、アプリケーションデバッガ（図示せず）によってアプリケーション６０のタスクが実行されたときに発行されるシステムコールを処理し、システムコールの処理によるＯＳ管理情報を変更する命令をＯＳ管理情報変更手段７２に通知する。ここで、ＯＳ管理情報とは、ＯＳシミュレータが管理するデータであり、例えばアプリケーション６０実行中のタスクの状態をあらわすタスク実行状況やタスクの状態を変化させるイベントの有無を１又は０のビットパターンで表したイベントフラグなどがある。

【００６５】

ＯＳ管理情報変更手段７２は、システムコール処理手段７１からのＯＳ管理情

報の変更命令に従って、後述する記憶手段１００に格納されているＯＳ管理情報Ｏ'Ｓ１を変更し、更にＯＳ管理情報ＯＳ１の変更内容を共有管理情報書込み手段６２に通知する。

【００６６】

共有管理情報書き込み手段６２は、ＯＳ管理情報変更手段７２から通知されたＯＳ管理情報ＯＳ１の変更内容を後述する共有ファイルＦ１に書き込む。

【００６７】

アプリケーション通信手段Ｃ１は、アプリケーション６０とＯＳデバッガ８０とのプロセス間通信を実行する。ＯＳ管理情報変更手段７２から通知されたＯＳ管理情報ＯＳ１の変更内容が共有管理情報書き込み手段６２によって共有ファイルＦ１に書き込まれたことをＯＳデバッガ８０に通信する。

【００６８】

記憶手段１００は、ＲＡＭ（Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ）で構成される。記憶手段１００には、ＯＳ管理情報変更手段７２で変更されるＯＳ管理情報ＯＳ１が記憶される。図３においては、一例として、アプリケーション６０のタスクＣに対応するＯＳ管理情報が記憶されている。ＯＳ管理情報ＯＳ１には、イベントフラグＥＦやタスク実行状態ＴＳが含まれている。イベントフラグＥＦはタスクＣの状態を変化させるイベントの有無を１又は０のビットパターンで表したものである。また、タスク実行状態ＴＳはアプリケーション６０実行中のタスクＣの状態を表したものである。図３の例では、タスクＣが実行状態であることが示されている。

【００６９】

共有ファイルＦ１は、他の実施の形態の共有ファイルＦと同様にアプリケーション６０とＯＳデバッガ８０とで共有されている。共有ファイルＦ１には、記憶手段１００に記憶されたＯＳ管理情報と同一のデータが含まれたデータである共有管理情報ＳＳ１が格納される。図３においては、一例として、アプリケーション６０のタスクＣに対応するＯＳ管理情報と同一のデータを含む共有管理情報ＳＳ１が記憶されている。共有管理情報ＳＳ１には、イベントフラグＥＦ１やタスク実行状態ＴＳ１が含まれている。イベントフラグＥＦ１は記憶手段のイベント

フラグＥＦに対応する。また、タスク実行状態ＴＳ１は記憶手段のタスク実行状態ＴＳに対応する。

【００７０】

また、共有ファイルＦ１の共有管理情報ＳＳ１には、アプリケーション６０のタスク毎の実行に伴う、ＯＳシミュレータ７０によるＯＳ管理情報ＯＳ１の変更に同期してＯＳ管理情報ＯＳ１の変更が反映されるようになっている。更に、アプリケーション通信手段Ｃ１からの変更通知をＯＳデバッガ８０が受信した場合に、ＯＳ管理情報ＯＳ１の変更が反映された共有管理情報ＳＳ１がＯＳデバッガ８０によって読み出されるようになっている。つまり、記憶手段１００のＯＳ管理情報ＯＳ１と共有ファイルＦ１の共有管理情報ＳＳ１とは完全に同期される。

【００７１】

ＯＳ管理情報を操作するためのユーティリティプログラムであるＯＳデバッガ８０は、ユーザーインターフェイス８１、メイン処理手段８２、共有管理情報読み出し手段８３、表示処理手段８４及びＯＳデバッガ通信手段Ｃ２とを備えている。

【００７２】

ＯＳデバッガ通信手段Ｃ２はアプリケーション６０とのプロセス間通信を実行する。アプリケーション通信手段Ｃ１からＯＳ管理情報ＯＳ１の変更内容が共有ファイルＦ１の共有管理情報ＳＳ１に書き込まれたという通知を受信した場合に、共有管理情報読み出し手段８３に共有管理情報ＳＳ１を読み出す命令をする。

【００７３】

共有管理情報読み出し手段８３は、ＯＳデバッガ通信手段Ｃ２の命令により、共有管理情報ＳＳ１を読み出し、読み出された共有管理情報ＳＳ１を表示するよう表示処理手段８４へ命令する。表示処理手段８４は共有管理情報読み出し手段８３によって読み出された共有管理情報ＳＳ１を表示制御部９１で映像信号に変換してディスプレイの画面（図示せず）に表示する。

【００７４】

ユーザーインターフェイス８１は、ユーザによって操作部（図示せず）からの入力信号がインターフェイス部９２を介して通知された場合、共有ファイルＦ１

に格納された共有管理情報ＳＳ１を表示させる処理をメイン処理手段８２へ命令する。メイン処理手段８２は、ユーザーインターフェイス８１の命令により、共有管理情報読み出し手段８３へ共有管理情報ＳＳ１を読み出す命令をする。

【００７５】

ここで、本実施の形態では、図示しない中央演算処理装置（ＣＰＵ）を含む制御部が、本発明のプログラムにしたがってＣＰＵ以外の各回路と協働して実行される制御動作を実現している。しかし、以下では説明の便宜上、ＣＰＵが関係する制御動作を、アプリケーション６０などのプログラムが直接的に制御しているものとしてその説明を行う。

【００７６】

以下に、アプリケーション６０、ＯＳシミュレータ７０及びＯＳデバッガ８０が備える各手段の具体的な処理の流れを図面を用いて説明する。図４は、デバッグ支援装置１の各手段の処理の流れを示すフローチャートである。

【００７７】

ここでは、例として、タスクＣが実行され、システムコールである「ｓｅｔ　ｆｌｇ」が発行される場合を用いて説明する。イベントフラグの任意のビットがセットされることにより、他のタスクを待ち状態としたり、待ち状態を解除して実行可能状態としたりすることができる。システムコール「ｓｅｔ　ｆｌｇ」が発行されるとＯＳ管理情報ＯＳ１のイベントフラグＥＦの値が０から１に変更される。システムコール「ｃｌｒ＿ｆｌｇ」が発行されるとＯＳ管理情報ＯＳ１のイベントフラグＥＦの値が１から０に変更されるようになっている。

【００７８】

まず、アプリケーション６０側の処理の流れを説明する。アプリケーションデバッガ（図示せず）によってアプリケーション６０のタスクＣが実行されて、システムコール「ｓｅｔ＿ｆｌｇ」が発行される（ステップＳ６０）。発行されたシステムコール「ｓｅｔ＿ｆｌｇ」がシステムコール処理手段７１で処理され（ステップＳ６１）、ＯＳ管理情報ＯＳ１のイベントフラグＥＦの値を０から１に変更する命令がＯＳ管理情報変更手段７２に通知される。

【００７９】

続いて、ＯＳ管理情報変更手段７２は、ＯＳ管理情報ＯＳ１のイベントフラグＥＦの値を０から１に変更し（ステップＳ６２）、イベントフラグＥＦを０から１に変更した内容を共有管理情報書き込み手段６２に通知する。

【００８０】

共有管理情報書込み手段６２は、ＯＳ管理情報ＯＳ１の変更内容を共有ファイルＦ１のイベントフラグＥＦ１に書き込む（ステップＳ６３）。具体的には、共有管理情報ＳＳ１のイベントフラグＥＦ１の値を０から１に変更する。

【００８１】

共有管理情報ＳＳ１のイベントフラグＥＦ１が変更された場合、アプリケーション通信手段Ｃ１は、共有管理情報ＳＳ１の変更された共有管理情報ＳＳ１を読み出す指示（以下、変更通知）をＯＳデバッガ通信手段Ｃ２へ送信する（ステップＳ６４）。その後、アプリケーション通信手段Ｃ２によって、アプリケーション６０の動作が停止される（ステップＳ６５）。ここで、アプリケーション通信手段Ｃ１からＯＳデバッガ通信手段Ｃ２への変更通知は、共有管理情報ＳＳ１が変更されたことを示すＲＡＭ上のフラグを介して実行してもよいし、ＯＳの通信機能を利用してもよい。

【００８２】

アプリケーション６０側では、アプリケーション６０の実行が停止されている間、アプリケーション通信手段Ｃ１はＯＳデバッガ通信手段Ｃ２からの返信があるか否かを判断する（ステップＳ６６）。ＯＳデバッガ通信手段Ｃ２からの返信がなければ（ステップＳ６６；Ｎｏ）、アプリケーション６０の動作は再開されず待ち状態となる。つまり、ＯＳデバッガ通信手段Ｃ２からの返信があるまでアプリケーション６０の次のタスクが実行されず、待ち状態が継続される。

【００８３】

続いて、ＯＳデバッガ８０側の処理の流れを説明する。ＯＳデバッガ通信手段Ｃ２は、ステップＳ６４でアプリケーション通信手段Ｃ１から送信された変更通知を受信する（ステップＳ８１）。ＯＳデバッガ通信手段Ｃ２は、共有管理情報読み出し手段８３へ共有管理情報ＳＳ１のイベントフラグＥＦ１を読み出すように命令する。

【００８４】

共有管理情報読み出し手段８３は、ＯＳデバッガ通信手段Ｃ２からの命令に従って共有管理情報ＳＳ１のイベントフラグＥＦ１を読み出す（ステップＳ８２）。読み出されたイベントフラグＥＦ１の情報は共有管理情報読み出し手段８３から表示処理手段８４へ通知され、表示制御部９１で映像信号に変更されてディスプレイの画面（図示せず）に表示される（ステップＳ８３）。

【００８５】

この際、イベントフラグＥＦ１が１であることをユーザが視覚的に理解できるように画面上に表示させることによって、ユーザはアプリケーション６０とＯＳ（図示せず）との処理の連携、アプリケーション６０とＯＳデバッガ８０との処理間で同期がとられたことを容易に把握することができる。

【００８６】

イベントフラグＥＦ１が表示された後、ＯＳデバッガ通信手段Ｃ２はアプリケーション６０を再起動する指示をアプリケーション通信手段Ｃ１へ返信する（ステップＳ８４）。ここで、ＯＳデバッガ通信手段Ｃ２からアプリケーション通信手段Ｃ１への返信は、共有管理情報ＳＳ１のイベントフラグＥＦ１が表示されたことを示すＲＡＭ上のフラグを介して実行してもよいし、ＯＳの通信機能を利用してもよい。

【００８７】

続いて、アプリケーション６０側の処理の流れに説明を戻す。アプリケーション通信手段Ｃ１はＯＳデバッガ通信手段Ｃ２からの返信を受信した場合（ステップＳ６６；Ｙｅｓ）、アプリケーションの動作が再開される（ステップＳ６７）。その後、再びステップＳ６０に戻りアプリケーションデバッガによって、タスクＣの次のタスクＤが実行されて、ステップＳ６０からステップＳ６７の処理が繰り返される。

【００８８】

アプリケーション通信手段Ｃ１とＯＳデバッガ通信手段Ｃ２との間で双方向で通信が実行されるまでアプリケーション６０の動作が停止されるので、ＯＳデバッガ８０の共有管理情報読み込み手段８３によって共有管理情報ＳＳ１が読み込

まれるまでＯＳシミュレータ７０によってＯＳ管理情報ＯＳ１が変更されることがなくなる。これにより、ＯＳデバッガ８０が、ＯＳ管理情報ＯＳ１の変更を取りこぼしなく取得することができ、アプリケーション６０と別プロセス空間に存在するＯＳデバッガ８０で、アプリケーション６０に同期してデータを共有することができる。

【００８９】

以上、システムコール「ｓｅｔ＿ｆｌｇ」の場合について説明したが、システムコール「ｃｌｒ＿ｆｌｇ」の場合でも同様にＯＳ管理情報ＯＳ１のイベントフラグＥＦが１から０に変更されたことをユーザに視覚的に伝えることができる。

【００９０】

また、現時点で実行されているタスク（以下、自タスク）の実行状態を変更する場合にも図４に示す処理を適用できる。例えば、自タスクＣを「実行状態」から「待ち状態」にするシステムコール「ｓｌｐ＿ｔｓｋ」、自タスクＣ以外のタスクやタスク以外が発行したシステムコールであり、自タスクを「待ち状態」から「実行可能状態」にするシステムコール「ｗｕｐ＿ｔｓｋ」の場合について説明する。

【００９１】

例えば、システムコール「ｓｌｐ＿ｔｓｋ」がシステムコール処理手段７１で処理された後（ステップＳ６１）、ＯＳ管理情報ＯＳ１のタスク実行状態ＴＳが「実行状態」から「待ち状態」に変更される（ステップＳ６２）。次に、共有ファイルＦ１のタスク実行状態ＴＳ１が、同様に「実行状態」から「待ち状態」に変更される（ステップＳ６４）。

【００９２】

アプリケーション通信手段Ｃ１からの変更通知がＯＳデバッガ通信手段Ｃ２へ送信された後（ステップＳ６５）、アプリケーション６０は待ち状態となる（ステップＳ６６）。ＯＳデバッガ８０側では、ＯＳデバッガ通信手段Ｃ２がアプリケーション通信手段Ｃ１からの変更通知を受信した後（ステップＳ８１）、共有ファイルＦ１のタスク実行状態ＴＳ１が読み出されてディスプレイの画面に表示される(ステップＳ８３)。

【００９３】

次に、ＯＳデバッガ通信手段Ｃ２は、アプリケーション通信手段Ｃ１へ返信し（ステップＳ８４）、アプリケーション通信手段Ｃ１で返信が受信された場合（ステップＳ６６；Ｙｅｓ）、アプリケーション６０の動作が再開される（ステップＳ６７）。

【００９４】

以上、システムコール「ｓｌｐ＿ｔｓｋ」の場合について説明したが、システムコール「ｗｕｐ＿ｔｓｋ」の場合でも同様にＯＳ管理情報ＯＳ１のタスク実行状態ＴＳが「待ち状態」から「実行可能状態」に変更されたことをユーザに視覚的に伝えることができる。

【００９５】

また、１つの自タスクの実行状態の変化を対象にしたが、アプリケーション６０の全てのタスクに対応するタスク実行状態を共有ファイルＦ１の共有管理情報ＳＳ１に記憶しておいてもよい。これにより、全てのタスクの状態をユーザが視覚的に把握することができるようになる。

【００９６】

また、図４の処理の流れの説明においては、ＯＳシミュレータ７０によるＯＳ管理情報ＯＳ１の変更に同期して、共有ファイルＦ１の共有管理情報ＳＳ１が変更されるごとにその変更された共有管理情報ＳＳ１が画面に表示されるようになっていたが、適宜変更可能である。

【００９７】

例えば、共有管理情報ＳＳ１の変更履歴をタイマを用いて周期的に共有ファイルＳＳ１に記憶しておき、ユーザからの指示があった場合にディスプレイの画面に表示するようにしてもよい。具体的には、ユーザがＯＳ管理情報ＯＳ１の変更をディスプレイの画面上で確認する場合、ユーザによってキーボードなどの操作部が操作されると共有ファイルＦ１に記憶された変更履歴が画面上に読み出されるようにすればよい。この場合、操作部から入力された入力信号がインターフェイス部９２を介してユーザーインターフェイス８１に通知され、メイン処理手段８３の指示にしたがって共有管理情報読み込み手段８３が共有ファイルＦ１から

変更履歴を読み出すようにすればよい。

【００９８】

例えば、移動体端末としての機能を実現するアプリケーション６０では、通話、電話帳登録、インターネット通信など様々な種類のイベント処理が考えられる。このような複数のイベントが連続的に処理されるリアルタイム処理が要求されるアプリケーション６０においては、全てのタスク間での状態遷移の相関を認識する必要がある。このような場合に、全てのタスクに対応する共有管理情報ＳＳ１の変更履歴を共有ファイルＦ１に記憶しておけば、全てのタスク間の状態遷移やイベントフラグなどの履歴や、アプリケーション６０とＯＳとの連携処理におけるエラー等を画面上で容易にユーザが確認できるので、デバッグ作業の効率が向上される。

【００９９】

以下に、共有ファイルＦ１から複数のタスクの状態遷移が画面に表示された例について説明する。図５は、複数のタスクの状態遷移が表示された画面の一例を示す図である。

【０１００】

図５において、ＤＰはディスプレイの画面を示す。ＳＣは、アプリケーション通信手段Ｃ１とＯＳデバッガ通信手段Ｃ２との間で通信が実行されたことを表すマーク群である。ＳＴ１は、イベントＡに対応するタスクの状態遷移を表すマーク群である。以下同様に、ＳＴ２はイベントＢに対応するタスクの状態遷移を、ＳＴ３はイベントＣに対応するタスクの状態遷移を表すマーク群である。

【０１０１】

各タスクの状態遷移ＳＴ１、ＳＴ２、ＳＴ３中のマークＭ１は、イベントに対応するタスクが「実行可能状態」であったことを示す。つまり、ＯＳシミュレータ７０によるＯＳ管理情報ＯＳ１中のタスク実行状態の変更に応じて、共有ファイルＦ１の共有管理情報ＳＳ１中のタスク実行状態が「実行可能状態」に変更されたことを示すマークである。また、各タスクの状態遷移ＳＴ１、ＳＴ２、ＳＴ３中のマークＭ２は、イベントに対応するタスクが「待ち状態」であったことを示す。つまり、ＯＳシミュレータ７０によるＯＳ管理情報ＯＳ１中のタスク実行

状態の変更に応じて、共有ファイルＦ１の共有管理情報ＳＳ１中のタスク実行状態が「待ち状態」に変更されたことを示す。

【０１０２】

ＳＣ１は、ＯＳシミュレータ７０によるＯＳ管理情報ＯＳ１中のタスク実行状態の変更に応じて、共有ファイルＦ１の共有管理情報ＳＳ１中のタスク実行状態が「実行可能状態」に変更され、アプリケーション通信手段Ｃ１とＯＳデバッガＣ２との間で通信が実行されたことを示すマークである。

【０１０３】

図５に示す例では、各イベントごとにそれぞれ、タスクの実行状態遷移、アプリケーション通信手段Ｃ１とＯＳデバッガＣ２との間の通信履歴とが図５中に示す時間軸に沿って表示されている。例えば、イベントＡの最初の５つのタスクは「実行可能状態」であり、イベントＢ、Ｃの最初の５つのタスクは「待ち状態」である。また、その間、アプリケーション通信手段Ｃ１とＯＳデバッガＣ２との間で通信が確実に実行され、アプリケーション６０側によるＯＳ管理情報ＯＳ１の変更を、ＯＳデバッガ７０が同期して読み込むことができたことが示されている。

【０１０４】

つまり、図５に示す画面例では、イベントＡ、Ｂ、Ｃにおける、アプリケーション通信手段Ｃ１とＯＳデバッガＣ２との通信履歴、各イベントのタスクの実行状態遷移が視覚的に示されている。これにより、アプリケーション６０で実行される各イベントの状態遷移、ＯＳとの連携などの不具合などをユーザは視覚的に容易に見極めることができるので、デバッグ作業の効率化を図ることが可能になる。

【０１０５】

また、ＯＳデバッガ８０によってタイマを用いて共有ファイルからデータを周期的に読み出す場合でも、共有ファイルからデータを読み出す周期と、ＯＳ管理情報がＯＳシミュレータによって変更される周期を確実に同期させることができる。

【０１０６】

・以上、説明した通り、アプリケーション通信手段Ｃ１とＯＳデバッガ通信手段C'2との間で双方向で通信が実行されるまでアプリケーション６０の動作が停止されるので、ＯＳデバッガ８０の共有管理情報読み込み手段８３によって共有管理情報ＳＳ１が読み込まれるまでＯＳシミュレータ７０によってＯＳ管理情報ＯＳ１が変更されることがなくなる。これにより、ＯＳデバッガ８０が、ＯＳ管理情報ＯＳ１の変更を取りこぼしなく取得することができ、アプリケーション６０と別プロセス空間に存在するＯＳデバッガ８０で、アプリケーション６０に同期してデータを確実に共有することができる。

【０１０７】

また、ＯＳ管理情報ＯＳ１の変更内容やその変更履歴を画面に表示することにより、アプリケーション６０やＯＳなどの動作を視覚的に把握できデバッグ作業の効率を向上させることができる。

【０１０８】

以上、本発明を具体的に説明したが、本発明は上記実施の形態に限定されるものではなく、その要旨を逸脱しない範囲で適宜変更可能であることはもちろんである。

【０１０９】

本発明に係るデバッグ支援装置が携帯電話などの移動体端末に適用された例を示したがこれに限定されるものではない。例えば、ネットワーク端末等、マルチタスク処理が要求されるリアルタイムＯＳが組み込まれた装置で実行されるアプリケーションのデバッグ作業に適用できる。

【０１１０】

また、ＯＳ管理情報の種別ごとにそれぞれ共有ファイルのメモリ領域を決めておけば、イベントフラグの値やタスクの状態だけでなく、セマフォの状態、メールボックスの状態などの他のＯＳ管理情報も同様な手順で変更履歴を表示させることができる。

【０１１１】

また、ＯＳシミュレータによって制御されるＯＳ管理情報を共有ファイル上に記憶し、記憶手段と共有ファイルとを共通化するようにしてもよい。具体的には

、ＯＳシミュレータが共有ファイル上のＯＳ管理情報の更新をすることにより、アプリケーションとＯＳデバッガとでＯＳ管理情報の共有することができるようになる。これにより、アプリケーションが停止した場合でもＯＳデバッガによってＯＳ管理情報が参照することができるようになる。

【０１１２】

さらに、本発明のプログラムをＣＤ－ＲＯＭ、ＦＤ、ＤＶＤ等の記録媒体に格納し、汎用コンピュータに対して着脱自在な形で提供される形態でもよい。これにより、本発明のプログラムが格納されたソフトウェア商品として装置と独立して容易に配布、販売することができるようになる。汎用コンピュータや汎用ゲーム装置などのハードウェアを用いてこのソフトウェアを使用することにより、これらのハードウェアで本発明が容易に実施できるようになる。

【０１１３】

加えて、本発明を実現するためのプログラムやデータは、通信回線などを介して接続された他の機器から受信してメモリに記録する形態であってもよい。さらには、通信回線などを介して接続された他の機器側のメモリに上記プログラムやデータを記録し、このプログラムやデータを通信回線などを介して使用する形態であってもよい。

【０１１４】

【発明の効果】

本発明によれば、ＯＳ管理情報の変更に同期して、ＯＳ管理情報の変更が共有ファイルの共有管理情報に反映されるので、アプリケーションが停止した場合でもＯＳ管理情報を容易に参照することができる。これにより、効率の良いデバッグを実行することができるようになる。

【０１１５】

また、ＯＳ管理情報をＯＳデバッガから変更できるため、ＯＳ管理情報が不正な値となった場合でも、アプリケーションの動作検証を継続することや、アプリケーションのプログラムを修正することなく割り込みが発生した状況をつくることが可能である。

【０１１６】

・更に、アプリケーションとＯＳデバッガとの間で双方向で通信が実行されるまでアプリケーションの動作が停止されるので、ＯＳデバッガが、ＯＳ管理情報の変更を取りこぼしなく取得することができる。

【図面の簡単な説明】

【図１】

本発明の第１の実施の形態におけるデバッグ支援装置の機能ブロック図

【図２】

本発明の第２の実施の形態におけるデバッグ支援装置の機能ブロック図

【図３】

本発明の第３の実施の形態におけるデバッグ支援装置の機能ブロック図

【図４】

同実施の形態に係るデバッグ支援装置の各手段の処理の流れを示すフローチャート

【図５】

同実施の形態にかかる複数のタスクの状態遷移が表示された画面の一例を示す図

【図６】

従来におけるデバッグ支援システムの概略機能ブロック図

【図７】

組み込み型アプリケーションの開発環境の説明図

【符号の説明】

１０　アプリケーション

１２　ＯＳ管理情報書き込み手段

１３　デバッグタスク起動手段

１４　システムコール情報生成手段

１５　デバッグ命令読み出し手段

１７　デバッグタスク

２０　ＯＳシミュレータ

２１　システムコール処理手段

２２　共有管理情報変更手段

３０　アプリケーションデバッガ

４０　ＯＳデバッガ

４１　ユーザインターフェイス

４２　メイン処理手段

４３　共有管理情報読み出し手段

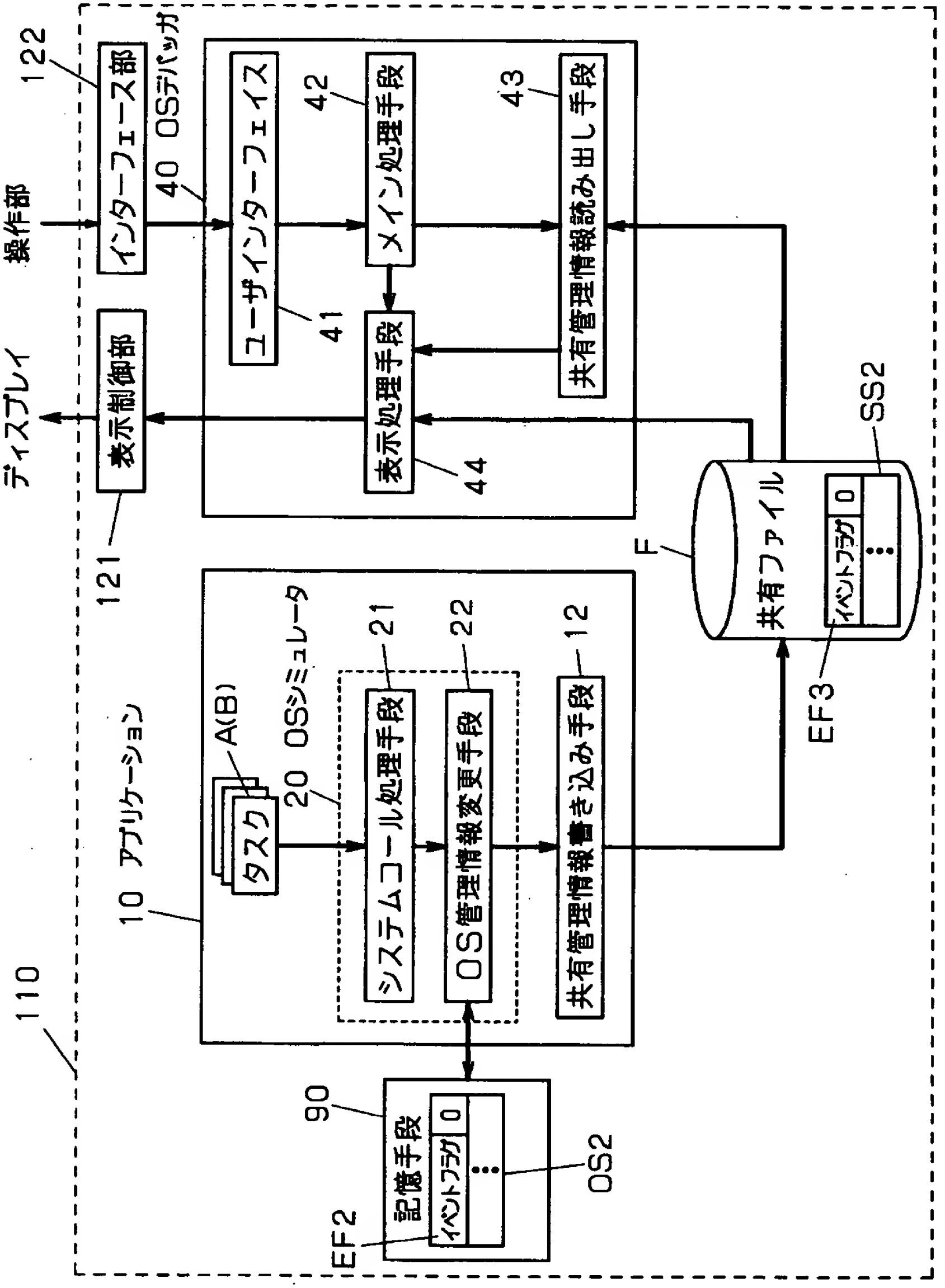
４４　表示処理手段

４５　デバッグ命令書き込み手段

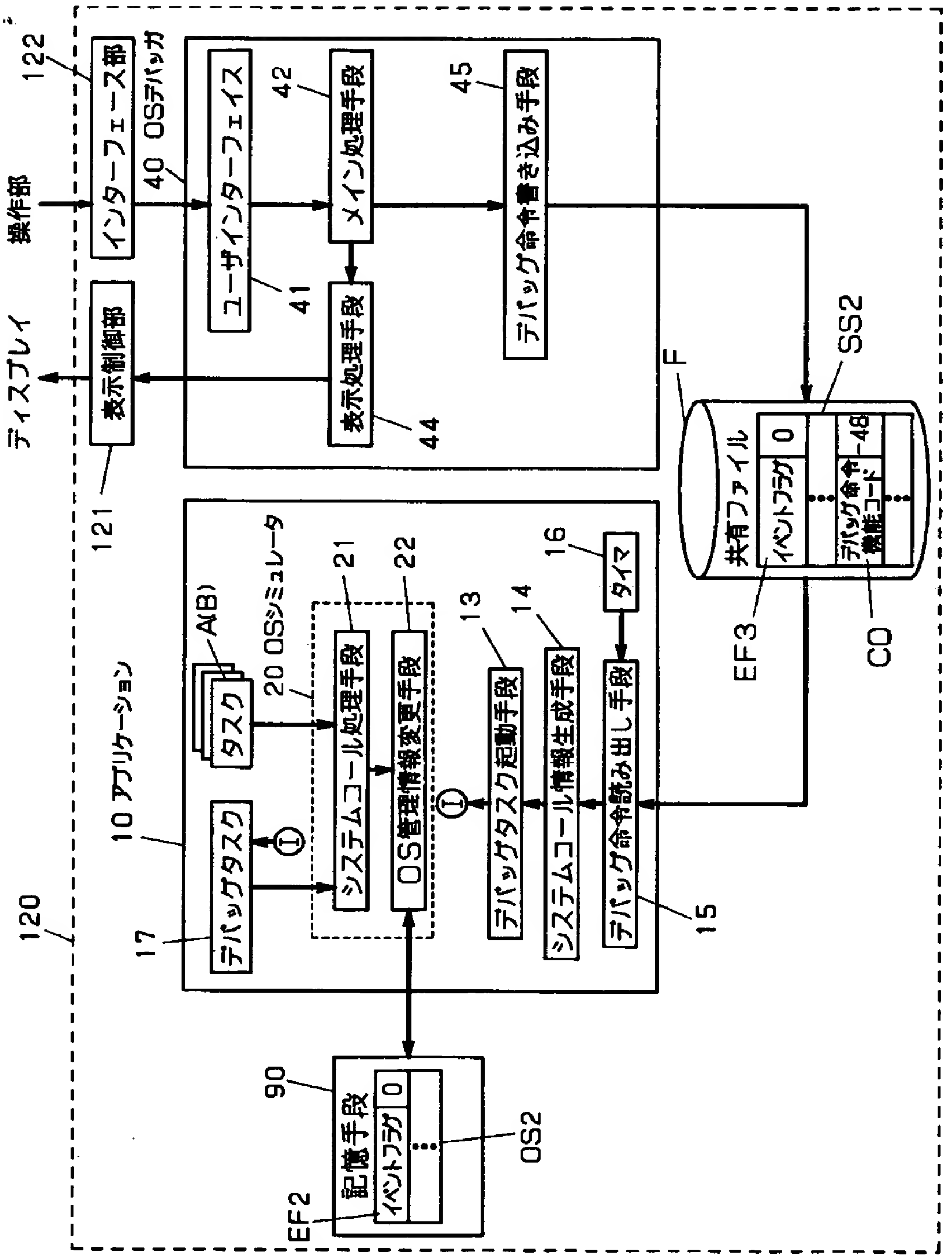
Ａ，Ｂ　タスク

【書類名】　　図面

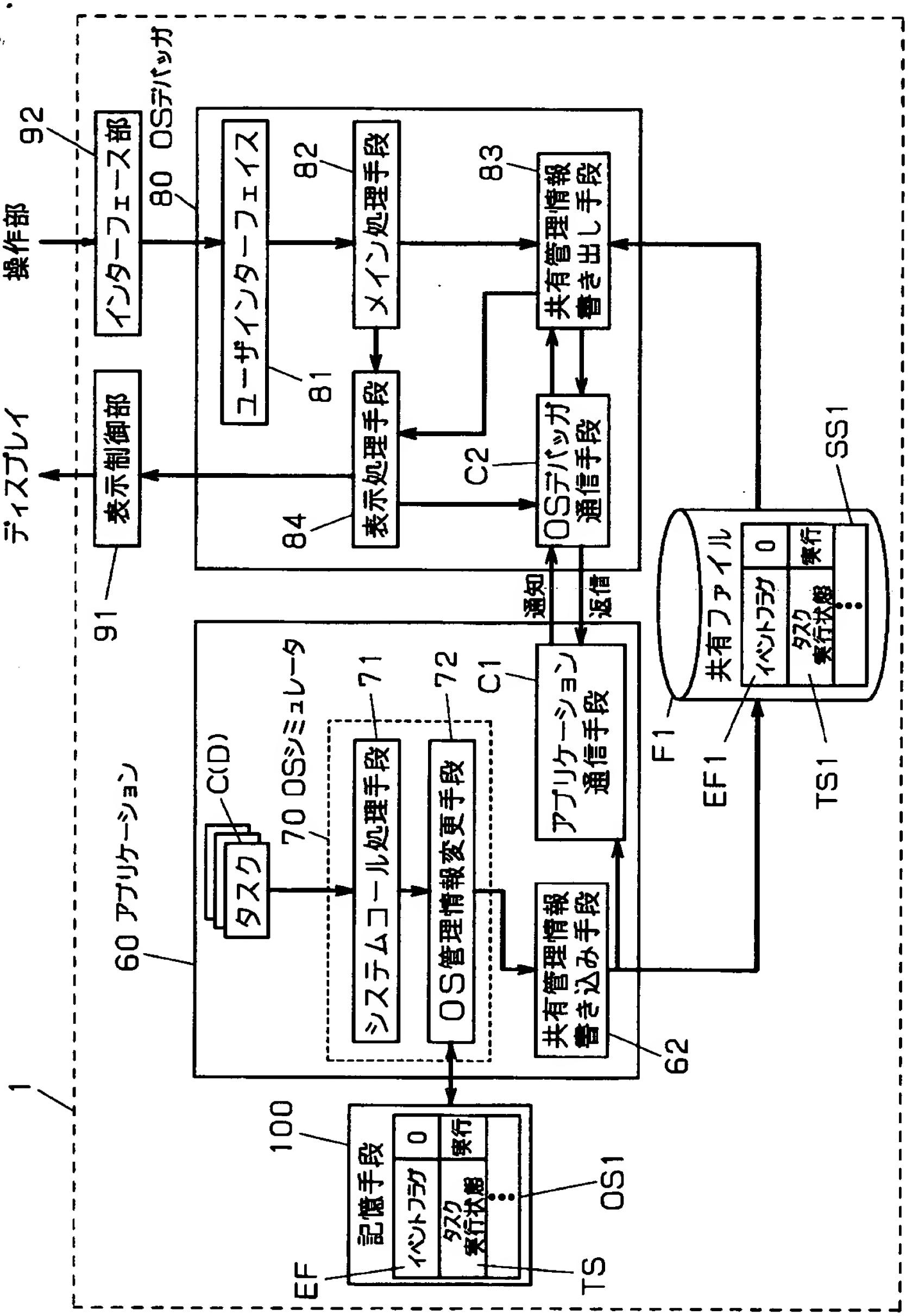
【図１】


操作部
ディスプレイ
122
インターフェース部
40 OSデバッガ
ユーザインターフェイス
41
42
メイン処理手段
43
共有管理情報読み出し手段
表示制御部
121
表示処理手段
44
共有管理情報書き込み手段
F
共有ファイル
イベントフラグ 0
SS2
EF3
10 アプリケーション
A(B)
タスク
20 OSシミュレータ
21
システムコール処理手段
22
OS管理情報変更手段
12
110
90
記憶手段
イベントフラグ 0
OS2
EF2

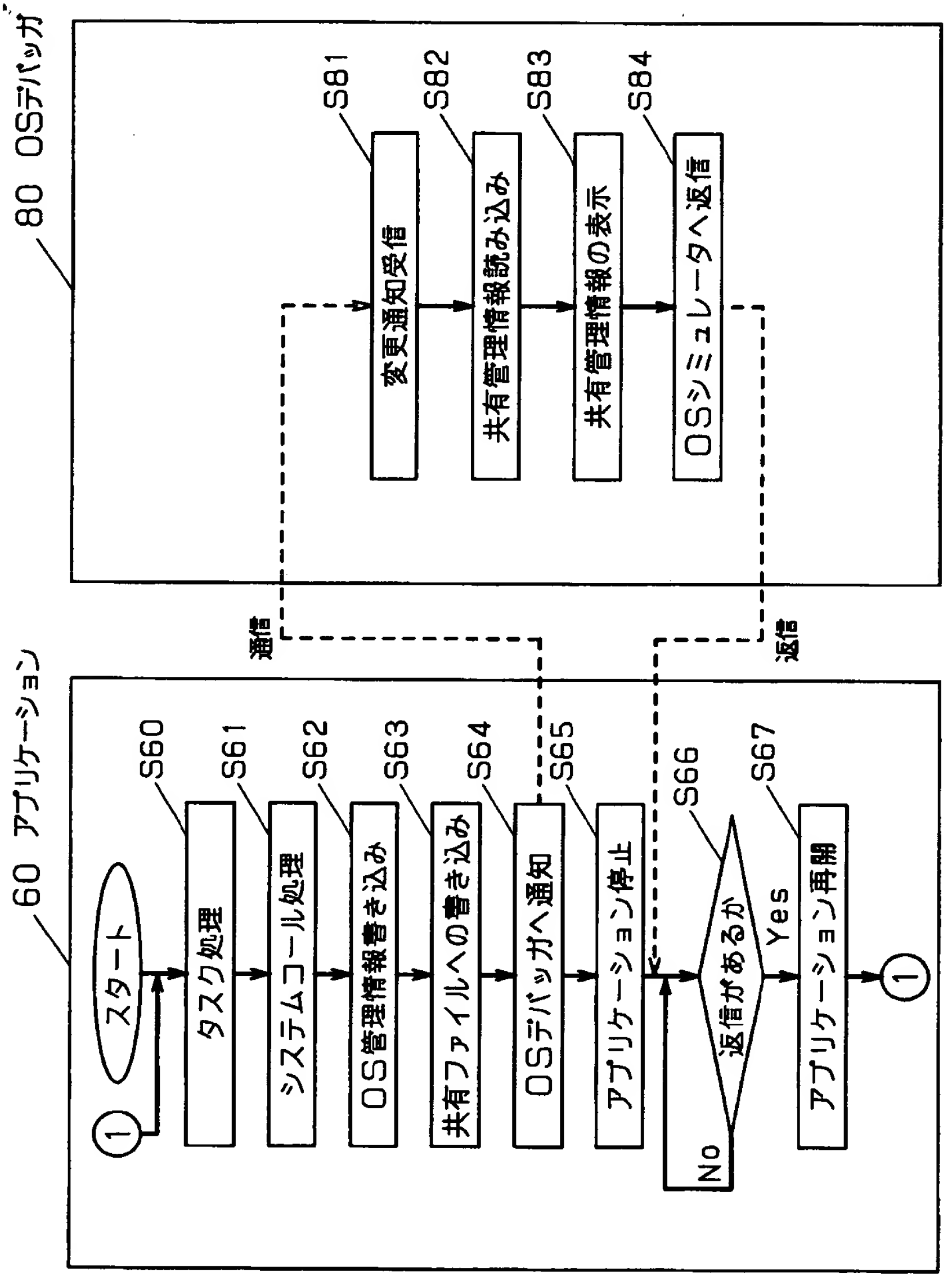
【図２】


120
操作部
122
インターフェース部
40 OSデバッガ
ユーザインターフェイス
41
42
メイン処理手段
45
デバッグ命令書き込み手段
ディスプレイ
表示制御部
121
表示処理手段
44
F
SS2
共有ファイル
イベントフラグ
0
デバッグ命令機能コード
48
EF3
C0
10 アプリケーション
A(B)
タスク
20 OSシミュレータ
21
システムコール処理手段
22
OS管理情報変更手段
16
タイマ
13
デバッグタスク起動手段
14
システムコール情報生成手段
デバッグタスク
17
デバッグ命令読み出し手段
15
90
記憶手段
イベントフラグ
0
OS2
EF2

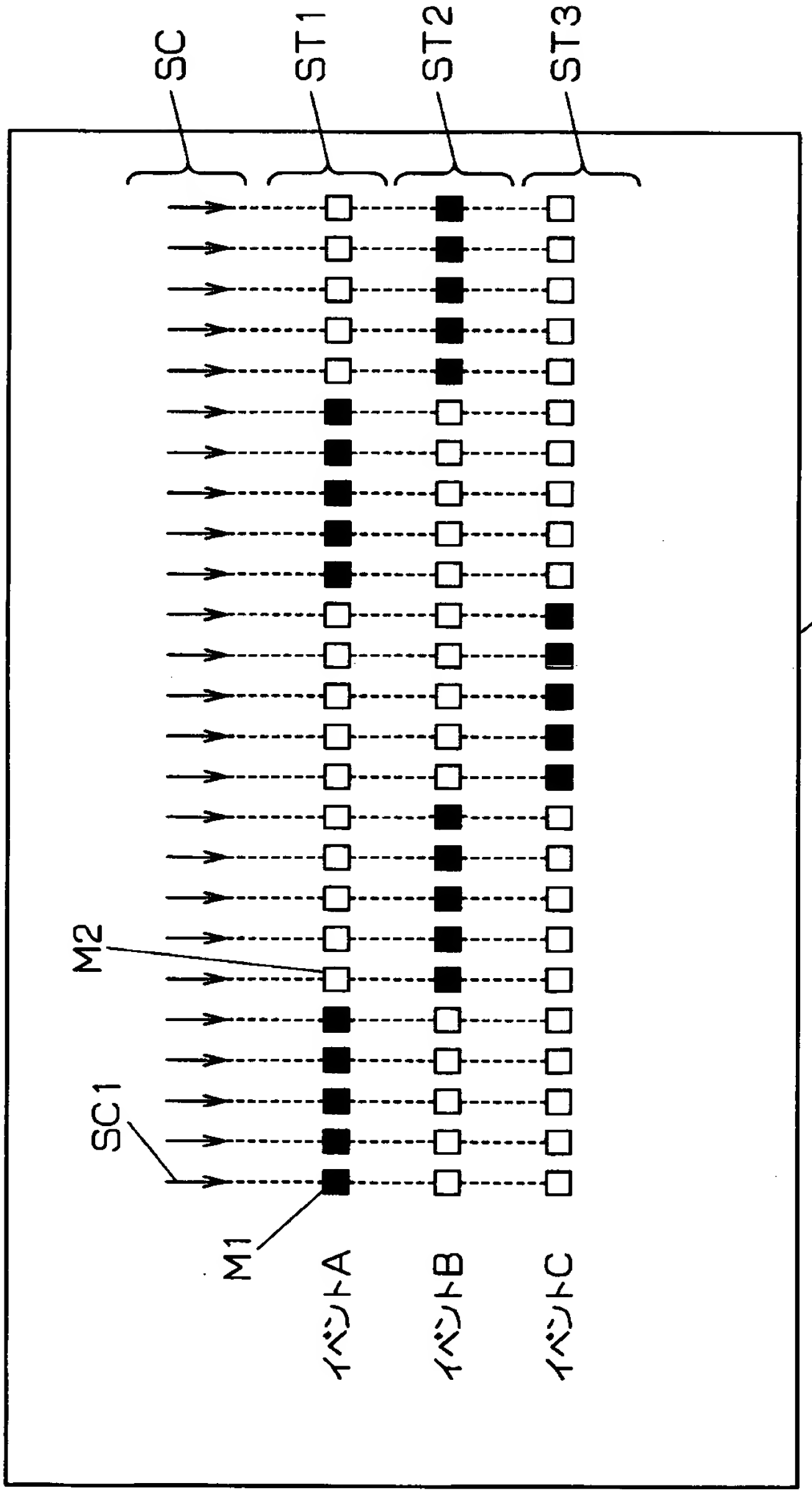
【図３】


操作部
92
インターフェース部
80 OSデバッガ
ユーザインターフェイス
82
メイン処理手段
83
共有管理情報書き出し手段
ディスプレイ
表示制御部
81
表示処理手段
C2
OSデバッガ通信手段
84
91
通知
返信
60 アプリケーション
C(D)
タスク
70 OSシミュレータ
71
システムコール処理手段
72
OS管理情報変更手段
C1
アプリケーション通信手段
共有管理情報書き込み手段
62
1
100
記憶手段
0
実行
イベントフラグ
タスク実行状態
EF
TS
OS1
F1
共有ファイル
EF1
TS1
SS1

【図４】

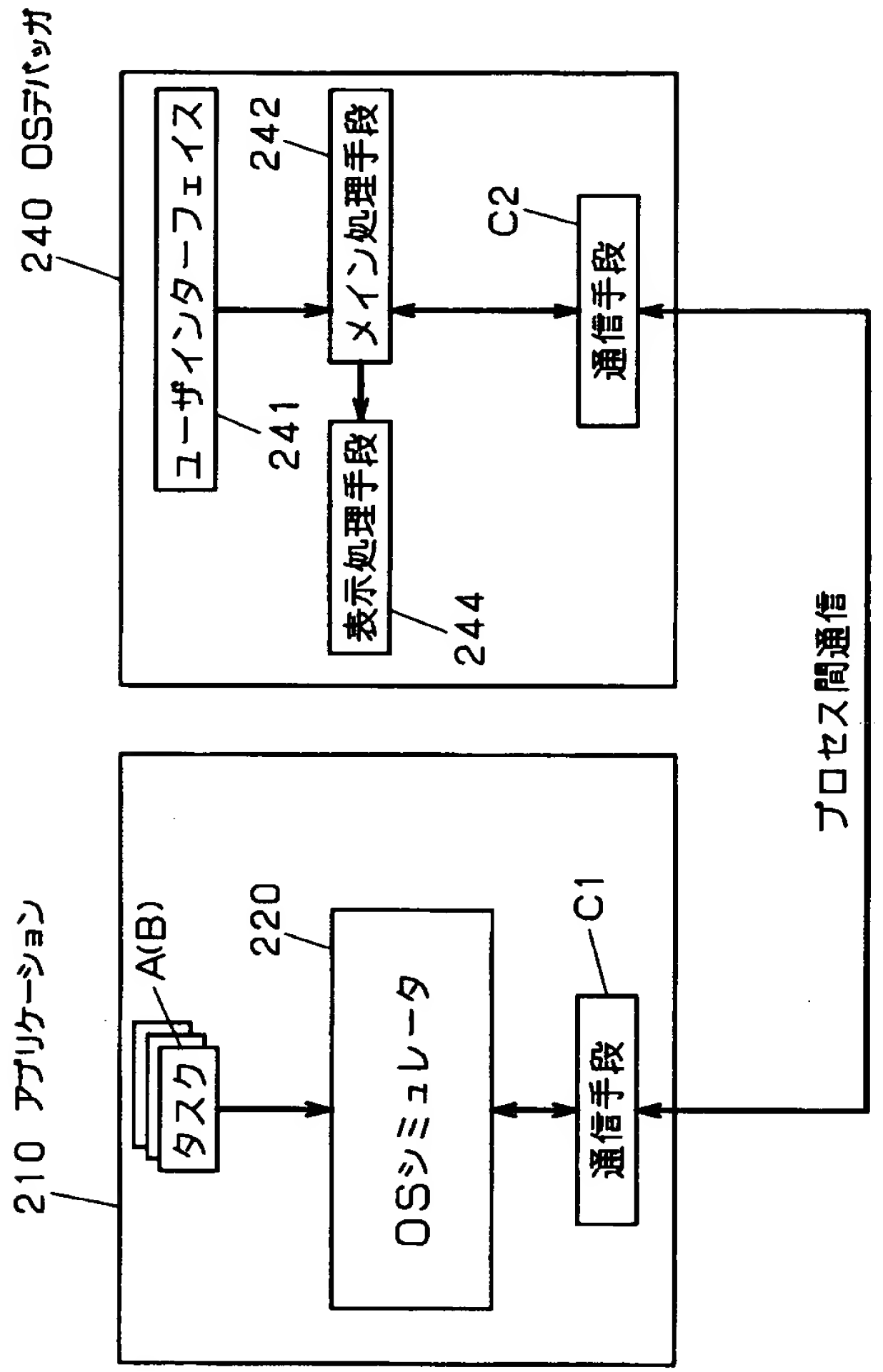

80 OSデバッガ
S81
変更通知受信
S82
共有管理情報読み込み
S83
共有管理情報の表示
S84
OSシミュレータへ返信
通信
返信
60 アプリケーション
スタート
S60
タスク処理
S61
システムコール処理
S62
OS管理情報書き込み
S63
共有ファイルへの書き込み
S64
OSデバッガへ通知
S65
アプリケーション停止
S66
返信があるか
No
Yes
S67
アプリケーション再開
①
①

【図５】

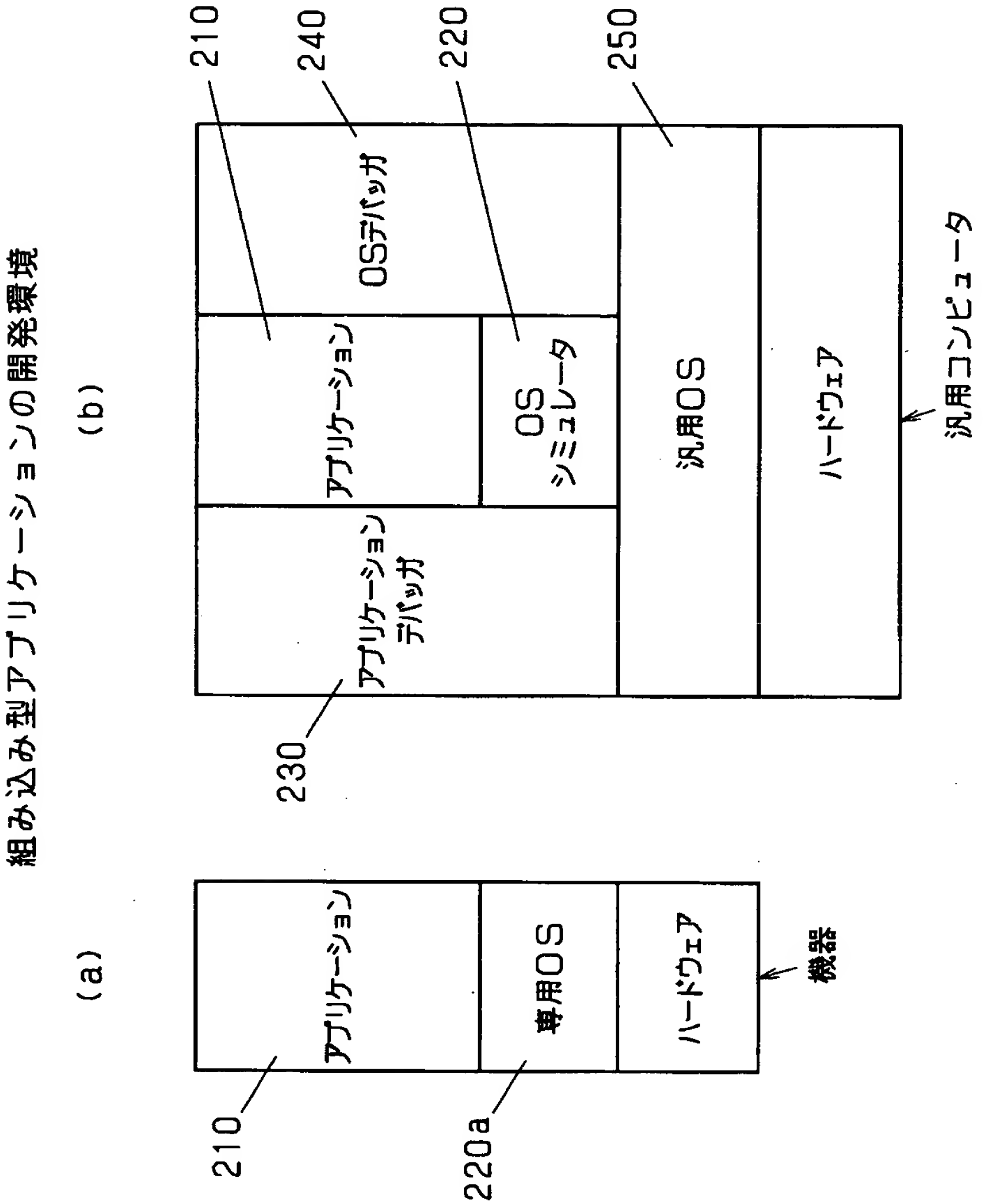


【図６】

従来におけるデバッグ支援システムの概略機能ブロック図



【図７】


組み込み型アプリケーションの開発環境
(a)
アプリケーション
専用OS
ハードウェア
機器
210
220a
(b)
210
240
220
250
OSデバッガ
アプリケーション
OS
シミュレータ
アプリケーション
デバッガ
230
汎用OS
ハードウェア
汎用コンピュータ

【書類名】　要約書

【要約】

【課題】　ＯＳシミュレータによるシミュレーション環境で開発したアプリケーションが停止した場合でも、この時点のＯＳ管理情報を参照できるようにしたデバッグ支援システムを提供する。

【解決手段】　ＯＳシミュレータがリンクされたアプリケーションと、リアルタイムＯＳの動作を仮想的に制御するＯＳデバッガとを有するシミュレーション環境において、アプリケーションとＯＳデバッガとで共有される共有ファイルに、ＯＳ管理情報と同一のデータを含む共有管理情報が記憶される。共有ファイルの共有管理情報へのＯＳ管理情報の書き込みや、書き込まれた共有管理情報を読み出すようにする。

【選択図】　図１

# 出　願　人　履　歴　情　報

識別番号　　〔０００００５８２１〕

１．変更年月日　　１９９０年　８月２８日
［変更理由］　新規登録
住　所　　大阪府門真市大字門真１００６番地
氏　名　　松下電器産業株式会社